CSC 261/461
Database Systems

Eustrat Zhupa

December 5, 2018

NOSQL Databases

Introduction

Companies and organizations are faced with applications that store vast amounts of data.

- e-mail application, Google Mail or Yahoo Mail:
  - can have millions of users
  - each user can have thousands of e-mail messages.
  - need a storage system that can manage all these e-mails
- a structured relational SQL system?
  - SQL systems offer too many services
  - a structured data model may be too restrictive

NOSQL Databases

Evolution

Organizations decided to develop their own systems:

- *Google* developed BigTable, used in many of Googles applications that require vast amounts of data stor- age, such as Gmail, Google Maps, and Website indexing.
- Apache Hbase is another system based on BigTable.
- Led to *column-based* stores.

NOSQL Databases

Evolution

- Amazon developed a system called DynamoDB, available in Amazons cloud services.
- Led to *key-value* data stores.
- Facebook developed a system called Cassandra
- Uses concepts from both key-value stores and column-based systems.
- MongoDB and CouchDB, classified as *document-based* systems.
- *Graph databases*: include Neo4J and GraphBase.

NOSQL Systems

Characteristics

- *Scalability*
  - horizontal: the distributed system is expanded by adding more nodes for data storage and processing as the volume of data grows.
  - vertical: expanding the storage and computing power of existing nodes.

NOSQL Systems

Characteristics

- *Availability*
  Many applications that use NOSQL systems require continuous system availability.
- *Replication*
  Data is replicated over two or more nodes, so that if one node fails, the data is still available on other nodes.

NOSQL Systems

Characteristics

Sharding of Files

- files can have many millions of records (or documents or objects)
- records can be accessed concurrently by thousands of users.
- it is not practical to store the whole file in one node.
- *Sharding* is a method for partitioning data across multiple machines.
- This serves to distribute the load of accessing the file records to multiple nodes.

NOSQL Systems

Characteristics

No Schema Required

- storing semi-structured, self-describing data.
- users can specify a partial schema, but it is not required
- any constraints on the data are coded.

Less Powerful Query Languages

- queries often locate single objects in a single file based on keys
- a set of functions and operations as API.
- operations are called (S)CRUD, for (Search), Create, Read, Update, and Delete.
- a high-level query language, but less power than SQL
- joins implemented in code.

MongoDB

MongoDB

Cross-platform, document oriented database providing:

- high performance
- high availability
- easy scalability.

MongoDB works on the concept of *collection* and *document*.

Database

Physical container for collections.

- Each database gets its own set of files on the file system.
- A single MongoDB server typically has multiple databases.

MongoDB

Collection

A group of MongoDB documents.

- Equivalent of an RDBMS table.
- A collection exists within a single database.
- Collections do not enforce a schema.
- Documents within a collection can have different fields.
- All documents in a collection are of similar or related purpose.

Document

A set of key-value pairs.

- Documents have dynamic schema.
- Documents in a collection do not need to have the same set of fields or structure,
- Common fields in a collection's documents may hold different types of data.

MongoDB vs RDBMS

The following shows the relationship of RDBMS terminology with MongoDB.

RDBMS	**MongoDB**
Database	Database
Table	Collection
Tuple/Row	Document
Column	Field
Table Join	Embedded Documents
Primary Key	Primary Key (Default key_id by mongodb itself)

MongoDB

Commands

- Start MongoDB

  ```
  sudo service mongodb start
  ```

- Stop MongoDB

  ```
  sudo service mongodb stop
  ```

- Restart MongoDB

  ```
  sudo service mongodb restart
  ```

- To use MongoDB run the following command

  ```
  mongo
  ```

  This will connect you to running MongoDB instance.

MongoDB

Example

A client needs a database for his blog/website

- Every post has the unique *title*, *description* and *url*.
- Every post can have one or more *tags*.
- Every post has the *name* of its publisher and total *number* of likes.
- Every post has *comments* given by users along with their *name*,*message*, *data-time* and *likes*.
- On each post, there can be zero or more comments.

MongoDB

Example

Design will have one collection post and the following structure

```
{
   _id: POST_ID
   title: TITLE_OF_POST,
   description: POST_DESCRIPTION,
   by: POST_BY,
   url: URL_OF_POST,
   tags: [TAG1, TAG2, TAG3],
   likes: TOTAL_LIKES,
   comments: [
      {
         user:'COMMENT_BY',
         message: TEXT,
         dateCreated: DATE_TIME,
         like: LIKES
      },
      {
         user:'COMMENT_BY',
         message: TEXT,
         dateCreated: DATE_TIME,
         like: LIKES
      }
   ]
```

MongoDB

Commands

- Create a new database if it doesn't exist

```
use DATABASE_NAME
```

- Use an existing database

```
use mydb
```

- To check your currently selected database

```
db
```

- To check your databases list

```
show dbs
```

MongoDB

Commands

- For the db to be visible you need to add data.

```
db.player.insert("name" :  "Di Biaggio")
```

- Default database is `test`.
- To delete a database

```
use mydb
db.dropDatabase()
```

MongoDB

Commands

To create a collection use

```
db.createCollection(name, options)
```

Field	Type	Description
capped	Boolean	(Optional) Enables a capped collection.
autoIndexId	Boolean	(Optional) Create index on _id field.
size	number	(Optional) Max size in bytes.
max	number	(Optional) Max number of documents.

MongoDB

Commands

- A collection without options

```
use test
db.createCollection("mycollection")
```

- To check collection in db

```
show collections
```

- A collection with options

```
db.createCollection("mycol", capped : true,
autoIndexId : true, size : 6142800, max :
10000 )
```

- Implicit collection creation

```
db.players.insert("name" : "Vialli")
```

MongoDB

Commands

- To drop a collection

  ```
  db.COLLECTION_NAME.drop()
  ```

- To drop the collection with the name mycollection.

  ```
  db.mycollection.drop()
  ```

- `drop()` returns true, if collection is dropped successfully, otherwise it will return false.

MongoDB

Commands

- To query data from collection

  ```
  db.COLLECTION_NAME.find()
  ```

- find() displays documents in a non-structured way.
- To display the results in a formatted way

  ```
  db.mycol.find().pretty()
  ```

- the `findOne()` method returns only one document.

MongoDB

Queries

RDBMS Where Clause Equivalents in MongoDB

Operation	Example	RDBMS
Eq	db.mycol.find({"by":"me"}).pretty()	where by = 'me'
LT	db.mycol.find({"likes":{$lt:50}}).pretty()	where likes < 50
LT Eq	db.mycol.find({"likes":{$lte:50}}).pretty()	where likes <= 50
GT	db.mycol.find({"likes":{$gt:50}}).pretty()	where likes > 50
GT Eq	db.mycol.find({"likes":{$gte:50}}).pretty()	where likes >= 50
Not Eq	db.mycol.find({"likes":{$ne:50}}).pretty()	where likes != 50

MongoDB

Queries

- More conditions with AND

```
db.mycol.find(
    {
        $and: [
            {key1: value1}, {key2:value2}
        ]
    }
).pretty()
```

- More conditions with OR

```
>db.mycol.find(
    {
        $or: [
            {key1: value1}, {key2:value2}
        ]
    }
).pretty()
```